

ELECTRONICALLY FILED 20 JUNE 2006

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of inventor(s):
Christoph Menzel et al.

Application No. **09/830,480**

Confirmation No. **8158**

Filing Date: **26 April 2001**

Title: **Internet Based Hearing Assessment
Methods**

Group Art Unit: **2153**

Examiner: **Aaron N. Strange**

CUSTOMER NO. 22470

MAIL STOP AMENDMENT

Commissioner for Patents

P.O. Box 1450

Alexandria, VA 22313-1450

DECLARATION

I, Ephram Cohen, declare as follows:

1. I am employed by the assignee of the above-identified patent application as Director of Software Engineering.

2. I have established dates for certain documents that are attached as Exhibits C, D and E attached to a COMBINED DECLARATION OF CHRISTOPH MENZEL, SUNIL PURIA, R. SCOTT RADER AND VINCENT PLUVINAGE.

3. Exhibits C and E are printed copies of electronic documents saved in Microsoft® Visual Source Safe®, which is a software program that maintains an audit trail for the documents that enables one to view changes to the documents and shows when the changes were made and by whom changes, if any, were made; and it is from these copies that I determined that Exhibit C was generated no later than February 2000 and Exhibit E was generated no later than April 2000.

4. Exhibit D is a printed copy of an electronic document saved employing Norton Ghost™, which enables generating backs up of information stored on a computer; and it is from this copy that I determined that Exhibit D was generated no later than March 2000.

The undersigned declares that all statements made herein of his own knowledge are true and that all statements made on information and belief are believed to be true; and further that the statements are made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of

Application No. 09/830,480

RXSD 1003-1

the United States Code and that such willful false statements may jeopardize the validity of the application and any patent that may issue therefrom.

DATE: 5/26/06

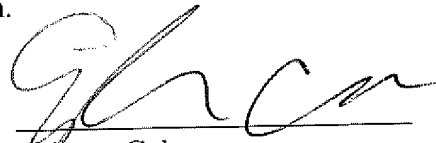

Ephram Cohen

EXHIBIT C

Document Revision History

REV	Change Description	Originator	Date

THESE DRAWINGS AND SPECIFICATIONS, AND THE DATA CONTAINED THEREIN, ARE THE EXCLUSIVE PROPERTY OF RxSOUND ISSUED IN STRICT

CONFIDENCE AND SHALL NOT, WITHOUT PRIOR WRITTEN PERMISSION OF RESOUND, BE REPRODUCED, COPIED OR USED FOR ANY PURPOSE WHATSOEVER, EXCEPT THE MANUFACTURE OF ARTICLES FOR RxSOUND

Name

Date

Original Author

APPROVALS

Approver

Approver 2



RxSound Corp,
801 Welch Road
Palo Alto, CA 94304

Document Title

AudioTest
Software Requirement Specification

Document Number

Xxxxxxxx70

Revision

REV

AudioTest

Software Requirement Specification

Document Number: XXXXXXXX70
Revision: REV

1. INTRODUCTION	3
1.1 Purpose	3
1.2 References and Glossaries	3
1.3 Document Scope	3
2. SYSTEM OVERVIEW	4
2.1 System Description	4
2.2 Key Functions	4
2.3 Program Limitations	4
2.4 Program Future	4
2.5 The User	5
2.6 Prerequisites	5
3. FUNCTIONAL REQUIREMENTS	6
3.1 User Screen and Menu	6
3.1.1 The Audiogram Display:	7
3.1.2 The "Administer" Area:	7
3.1.3 The options area:	7
4. INTERFACE REQUIREMENTS	8
4.1 User Interface	8
4.2 Hardware	8
4.3 Communications Interface	8
4.4 Software Interface	8
5. OTHER REQUIREMENTS	9
5.1 Timing (Real time Requirements)	9
5.2 Quality	9
5.3 Data logging	9
5.4 Outstanding Issues	9

1. Introduction

1.1 Purpose

AudioTest is a small application created to prove the concept of performing an audiometric test using a standard PC and calibrated head phones.

In its first implementation it will facilitate easy manual administration of tones to subjects along with recording of the results in some sort of datafile.

Please see chapter 2.4 with respect to future developments.

1.2 References and Glossaries

[ITHFS] Internet Hearing Test Functional Specification. Chris Menzel.

1.3 Document Scope

At this point the document is a work in progress to ensure that the developer (Benny) and the project manager (Chris) have a mutual understanding of what the program should be doing.

2. System Overview

This section describes the program and the environment in which it must work. It also presents a short list of the main features to be incorporated.

2.1 System Description

The AudioTest program is a standalone PC program. It communicates with the user through the screen, mouse and keyboard. It communicates with the sound card to produce various stimulus as output, and maybe obtain some sound level measurement as input.

2.2 Key Functions

The program has a number of user related functions:

- Calibration. The user can follow a step by step procedure to calibrate the system.
- Enter client information. A very simple set of fields allows the user to identify the patient under test.
- Administer test. This function allows the user to use the screen as a simple (yet flexible) audiometer.

2.3 Program Limitations

This development will NOT include an installation program, neither will help files be included. It is only an "in house" program.

For the first version, the user interface will be functional, but not really "fancy".

It is expected that the program can use standard Windows functions for controlling the sound card (if not it would be a nightmare keeping updated with different hardware configurations).

Also see section 4.4 – Software Interface.

2.4 Program Future

The program is only a "beginner program", intended to achieve two goals.

- Provide Chris a platform on which he can refine the hearing test and provide the audiologists a better tool to administer the test.
- Give Benny a first introduction to the problem domain.

Through gradual development, the following features are expected to be added:

- Automatic test administration. This includes implementation of one or several test algorithms.
- User interface improvement.
- Gradual componentization, so the core parts of the test can be used in several development environments. (Visual Basic, Java, JavaScript)

- Eventual transition so it can run on the web – on a yet to be determined set of platforms.

2.5 The User

The first version of the program will only be used internally in RxSound.

2.6 Prerequisites

None.

3. Functional Requirements

This lists the requirements from the users perspective.

3.1 User Screens and Menu(s)

There will only be one screen, looking as the one shown below. By accessing a menu, the user has the following options:

- New: Creates new file, ready for
- Open an existing file (which allow the selection of a datafile)
- Save. Save the measurement
- Save as.... Save the measurement under another file name.
- Calibrate.... Opens the calibration dialog.

3.2 The AudioTest main Screen

AudioTest (rev 1)

	125	250	500	750	1k	1k5	2k	3k	4k	6k	8k
R:	10	15	20	15	25	35	40	45	40	45	50
L:	30	35	35	40	40	40	50	50	60	60	65

Options

Test Signal:
Std Warble

Duration:
2 loops

Administer

☐ Right ☐ Left

Hz **dB**

250 65

SOUND

The AudioTest screen is divided into 3 sections:

3.2.1 The Audiogram Display:

This area shows the recorded values. It is read only, and will be updated as the test progresses.

3.2.2 The "Administer" Area:

These controls are used to administer the test. Using the up and down arrows the user can change the frequency or the sound pressure level. Pressing the largest button changes the sound pressure level in increments of 5. Pressing the smaller ones changes the level in increments of 3 and 1 dB respectively.

To initiate a signal the user will press the SOUND button, and a number of things may happen, depending of the settings of the "duration" field in the "options" section.

- If duration options is : "no loop" the sound will stop as soon as the button is released
- If duration options is: " x loops" the sound will sound for x loops of the sound file.
- If duration options is "indefinitely", the sound will continue until the button is pushed again.

3.2.3 The options area:

This should allow the "expert user" to try out a few things, that are then reflected, when the actual administration is performed.

Currently two options are suggested:

- Test signal. The user may select different types of test signals. What this boils down to in the end, is that different families of wave files are being selected. *QUESTION: Do we really want this here? Is it not a mess if part of the test is conducted using one signal, and other parts are conducted using another? So maybe it should not be possible to mix signals this easy.*
- Duration. This determines how long the signal is administered (also see above).

4. Interface Requirements

The following is a list of requirements to the user interface, as well as to interfaces required for this software to work.

4.1 User Interface

N/A

4.2 Hardware

At this point a PC with Windows 95 or above, and sufficient Sound Card capabilities as determined by Chris.

4.3 Communications Interface

N/A.

4.4 Software Interface

It is anticipated that the whole program can be written only using calls to standard Windows API, or maybe the DirectX library.

Chris will deliver to this application a number of different sound-files, in a yet to be determined format. AudioTest will be able to read these files and use them to generate the test stimulus after some minor modifications. Modifications include only:

- Attenuation
- Looping (a set amount of time)

(The whole sound generation will however be abstracted, so if things have to be done differently at a later stage, this should not affect the user interface or the test algorithm significantly)

5. Other Requirements

I am sure I can think about something.

5.1 *Timing (Real time Requirements)*

None – other than what is required to support the production of sound.

5.2 *Quality*

Please rank the following attributes in order of importance:

- At this point – development time is most critical.

5.3 *Data logging*

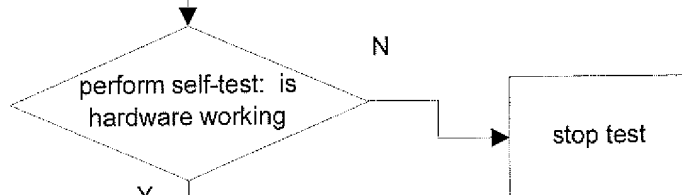
The result of every test will be logged in a .tst file. The format and contents of this file is still TBD.

5.4 *Outstanding Issues*

N/A

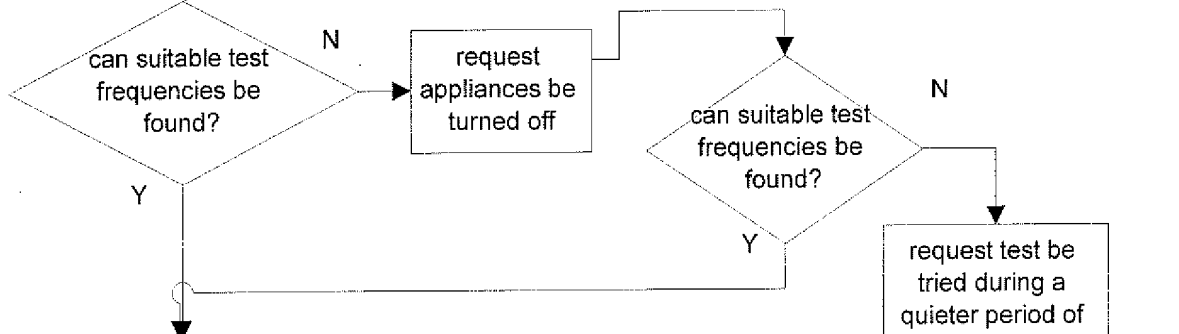
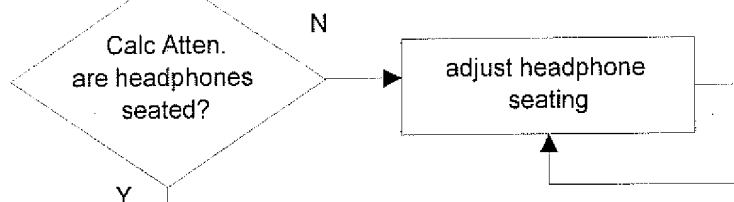
EXHIBIT D

calibrate A/D:
1) apply cal voltage to A/D
2) apply cal. voltage to transducer

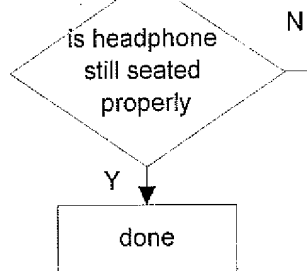


measure noise: headphones off

measure noise with headphones on.



perform threshold test flow



Notes

1) calibration of the A/D can proceed in one of two ways. If the transducer is factory calibrated, then the voltage is applied to the transducer. If the uphone is factory calibrated, the the voltage is applied to the uphone

2) suitable test frequencies are those with with a low enough 1/3 octave band noise

Some things we've discussed wrt to implementation:

Calibration of A/D. There are (at least) two methods to do this:

Method	What knows	Where is cal signal applied	comments
Uphone based	Uphone sensitivity and frequency response	Signal is a voltage, it is applied to the Sound card A/D	
Transducer based	transducer sensitivity and frequency response	Signal is a voltage, it is applied to the transducer	Full sound field generated must be known ie are headphones on head?, what reflections in room etc

Determination of which one is best will be related to:

- Robustness of calibrated component: how likely to change during shipping and on site
- Cost of calibration per component OR cost to perform calibration

Headphone leakage

This issue may be less of an issue than previously represented. We have a paper that shows that for holes leaks up to 1.65mm diameters, the effect of the leak has disappeared by about 400hz. (Anderson and Whittle, Acustica 1971). So it can be surmised that larger leaks will have little or no effect at 2K and above where we expect to make our measurements. Clearly some testing will need to be done on this issue.

Background noise level measurement

All of our estimates on what the background noise inside the headphones will be (and hence our smallest measureable hearing loss) are based on the noise present in a 1/3 octave band. Indeed, it is assumed that the total room noise will be more in the 50-60dB range. Hence, it is a necessary condition for us to actually measure the 1/3 octave band noise. This does two things:

- 1) changes how we look at the uphone noise specification ie we will need to know the 1/3 octave band noise as a function of center frequency
- 2) requires that spectral analysis capabilities are part of the system.

EXHIBIT E

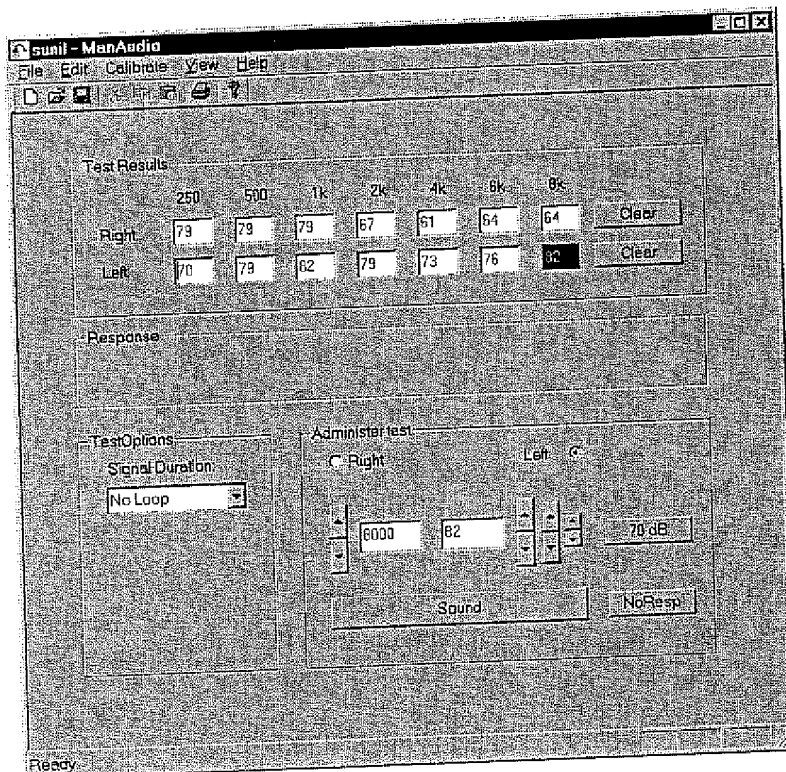
Memo

To: Scott Rader; Chris Menzel; Sunil Puria; John Winstead
From: Benny B. Johansen
CC:
Date: [REDACTED]
Re: Status and future plans: Update on Manual Audiometer Program

Purpose and Contents

I have now created the first visible program, which tests out some of the features which we need in the home audiotest and for future research. Below I will first mention where I am with the audio test, then I will outline my plans on the development front for the next couple of weeks.

Status: The ManAudio.Exe program



How to Find and Install?

Currently it is available for installation on RPA2\RXTRANSFER\MANAUDIO. Just run the setup program and follow the instructions.

What happens the first time/Calibration?

When you run it the first time, you will probably get an error. This is because the program relies on calibration data, which is specific to your sound card. To "calibrate" your system do the following:

1. Notice the key, which the program claims it is missing. It will have the format SC_XX_XX_XXXX
2. Connect the calibrated phones and locate the SPL meter over one of the loudspeakers.
3. Use the function Calibrate|1000Hz at peak output, and note the output on the SPL meter.
4. Add an entry: SC_XX_XX_XXXX= <measured value> to the filemanaudio.ini under the [SOUNDCARD] section: as shown below:

```
[SOUNDCARD_SPEC],
;SC_<ManufNo>_<ProductID>_<DriverNo>           =           <MaxOutput>
;ESS                                           Technologies....
SC_46_1_200=93
SC_46_2_200=0
SC_1_100_500=93
```

Close the program, and restart it, and your system should now be "calibrated".

Notice that if you later install ManAudio again this file will be overwritten. (Not nice, but it will take too much time to fix for now)

Functions to measure threshold:

The function is modeled an audiometer, and you should be able to use it without using the keyboard.

<key up>/<key down>: Level up or Level down in 3 db steps. Hold down the left <shift> key and you will increase and decrease by 6 dB. Hold down the right <shift key> and you will increase and decrease by 1 dB.

"7" Set level to 70 dB

"N" Mark "NoResponse"

<SPACE> Present Sound

The values shown on the screen are db HL.

If you locate the mouse on top of the field named "response", <right mouse clicking> will cause the "response field" to light up, indicating that the user could hear the stimulus. (An entry will also be written in the log file).

File Management:

Very simple. Use "New" to create a new test, and Save and Save as to save test results.

The result will be saved as an .aud file, and notice that this is a text logfile, that should be viewed using notepad or any other text editing program.

The LogFile:

The logfile is separated into the following sections:

#General	Standard information about the test.
#DATA	The actual measurements
#HARDWARE	Identification of the wave in and wave out hardware and software drivers
#LOG	A fairly elaborate log of the flow of events, including the users response.

What has been tested

At this point I have only installed the program on my own and Chris Menzels machine. It would be interesting to try it out on more machines, to get a good feel for the capabilities of the different sound cards and my ability to control them.

Which capabilities have been demonstrated?

- The overall architecture of different modules communicating via formatted text strings seems to work well. Likewise the logging seems to be useful.
- I am able to control the sound card as follows:
 - Play any wave file, implement looping and attenuation. Relate output to a calibration factor.
 - Capture wave input, simultaneously with playing wave output.
 - Increase the controls on the audio mixer when running the application, and resetting them once the application is completed. (Currently I am only doing it for waveout and overall volume out. I also know how to do it for microphone in, but it is a rather lengthy procedure, so I have left it out for now).

Which capabilities are (almost) also available?

- I have a separate program, which does FFT on the captured input signal. I just have not figured out how to calibrate the input. (Once I have, I think it would make sense to include another row of numbers on the audiometer showing the measured noise floor at each of the target frequencies).
- Same program also demonstrates a little graph utility which plots the results.
- Whether you select "left" or "right" the sound will be presented binaurally. I know how to change this to monaural presentation, I just have not had the time to do it.

An Interesting Observation:

Although the whole installation "disk" is currently 1.5 MB, it is interesting to notice that the program itself is only 76 KB. The rest is taken up by the wave files. Once we settle on the type of wavefile we want, it would be fairly easy to write to the code to generate those files internally, leaving the total program to be less than 100 KB, something which is certainly possible to download over the internet.

The next couple of Weeks:

I think the ManAudio application can be useful for testing a number of things with respect to sound cards, microphones etc. and I will continue to extend it, whenever it make sense.

However the ultimate demo, which I am working towards is the following:

- User accesses a web-page on our internal server a) using IE4 or IE5 b) maybe using Netscape 4.0 or higher.
- The page contains a hidden activex control and a single button, which can be used to indicate if a sound can be heard.
- Using this page, the user can take an interactive hearing test, wearing our calibrated head phones.
- Once the test is completed the results will be stored on our central database server.
- The user will then go to a second page showing a list of music selections.
- The user can pick one selection. This selection will immediately be processed in the back end by a PC with a DSP board.
- A new screen will then be shown to the user containing a link to the modified sound file.

I plan to take the following approach:

- Write a simple C++ program which implements the automated procedure. Have John Winstead try it on a few people.
- Change this simple C++ program into an Active X control and embed it into a web-page
- Have Scott find some DSP consultants to implement the compressing routine on a Texas Instruments DSP processor. If this cannot be done in time, I may just implement something very simple to demonstrate the principle, or "borrow" Brian Moores algorithm and run it on the server.
- Implement the link between the work station Active X control and the back end server.
- All work will be done on Internet Information Server 4, and using ASP + one of two COM objects.

With a lot of luck and some help I can probably accomplish this by the end of April, beginning of May.